

Concurrencia

Pau Arlandis Martinez

Sobre las normas

Profesores

- Angel Herranz – 2309
- Julio Mariño – 2308

Cada lunes se entregará un problema que debe resolverse antes del jueves. Únicamente sirven para practicar y para salvar a alguien que esté casi aprobado. 10 ejercicios.

Parte teórica → 50% (2 exámenes)

Parte práctica → 50% (2 partes) → En grupos de 2 personas.

Cada examen debe superarse con al menos un 4 para hacer media.

Tema 1

Ejercicio 0

Leer [Concepts and Notations for Concurrent Programming](#) de G.R. Andrews y F.B. Schneider (1983) hasta la sección 3.2 (incluida).

En esta asignatura (y en la informática en general) es bueno conocer los trabajos de tres nombres propios:

- Dijkstra
- Hoare
- Knuth

Puesto que crean toda la base teórica y conceptual de la informática.

En Concurrencia debemos tener dos conceptos de Java absolutamente claros:

- Scope o ámbito de variables.
- Variable vs Objeto.

Artículo recomendado de la semana sobre neurociencia y aprendizaje tecnológico: [Unskilled and unaware of it](#)

TODO desde página 1.2 a 4.1

Mecanismos de concurrencia

Semáforos

Es un TAD con dos operaciones atómicas, esperar y señalar:

```
class semáforo{
    private int cont = 0; //Contados, a 0 en este ejemplo.

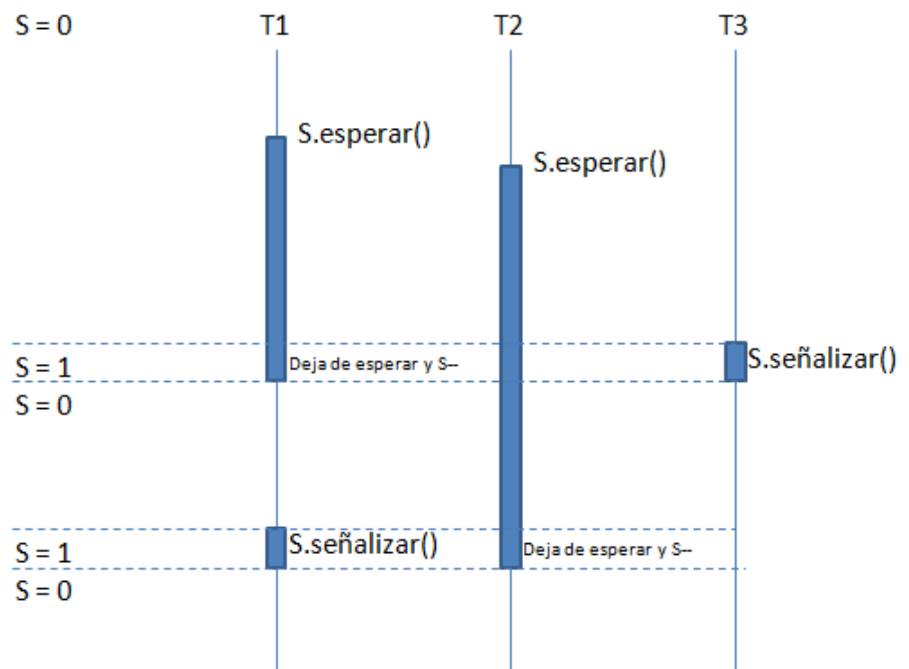
    public void esperar(){
        /** Espera hasta que cont > 0 y entonces cont-- */
    }

    public void señalar(){
        cont++; //Incrementa el contador.
    }
}
```

Para evitar condiciones de carrera cada operación de los semáforos está implementada atómicamente, esto quiere decir, que dos procesos no pueden ejecutar al mismo tiempo las operaciones de un semáforo. Si dos procesos tratan de efectuar la misma operación al mismo tiempo la efectuará primero la que llegue primero, sin sorpresas.

La utilidad de un semáforo es que cuando un proceso ejecute esperar() y el contador esté a 0, esperará hasta que otro proceso aumente el contador (al finalizar su sección crítica, por ejemplo).

La cuestión ahora es qué sucede cuando dos o más procesos están esperando y otro proceso ejecuta señalar. Lo que sucede es que un semáforo solo va a desbloquear a uno de los procesos, uno y solo uno; como puede verse en el diagrama. Siguiendo una propiedad de la concurrencia denominado Fairness (equitatividad).



Existen tres tipos de equitatividad:

- Equitatividad fuerte (FIFO). Garantiza equitatividad en el infinito.
- Equitatividad débil (Azar). Garantiza equitatividad en el infinito.
- Equitatividad "nula" (LIFO). No garantiza equitatividad en el infinito.

Los semáforos en las librerías

Cuando te encuentras ante una librería de semáforos debes mirar dos cosas básicas:

- Siempre están las operaciones esperar y señalar, hay que mirar como se denominan en esa librería. En Java: esperar = acquire y señalar = release.
- Hay que mirar cómo se construyen los semáforos. En Java: El constructor se llama con uno o dos parámetros:
 - Un contador (int) que será el número inicial del contador del semáforo. Parámetro obligatorio.
 - Un parámetro booleano que construirá el semáforo con equitatividad fuerte (si true) o sin equitatividad (a false). Es un parámetro opcional.

En la librería propia de la asignatura, CCLib, tenemos las siguientes características:

- El constructor ya no necesita parámetros (por defecto, int=0 y con equitatividad), aunque pueden darse.
- Esperar es await(). No necesita capturar la excepción y siempre decrementa el contador.
- Señalizar es signal().



Apuntes de Concurrencia by [Pau Arlandis Martínez](#) is licensed under a [Creative Commons Reconocimiento 3.0 Unported License](#).